1-17/2

Appendix A

Source Code Listing

```
% ROGPS – Solves for user position and time based on ephemeris equations
% and on time–difference measurements from 5 GPS satellites.

% U (3x1) the users position
% time (1x1) seconds
% Ub (3x1) an initial guess at the users position
% dU (3x1) is the adjustment made on the initial guess
% sat (3x5) is the location of each satellite
% H (4x4) matrix differentiating r_diff wrt (U,t)

% sat0 is the (3x5) matrix of 5 sat positions at beginning of
% interpolation interval.
% sat2 is the (3x5) matrix of 5 sat positions at end of interpolation interval.
% V is a (3x5) matrix of velocities for all 5 satellites.

  dU = 0;
  % step 1: Guess an initial Ubar, and time.
  U_true = [1300748.12;–4500694.5;4313770.5]; % Actual user location
  Ub =    [1200748.12;–4400694.5;4319770.5]; % User location guess

  time_true = 483738.75;
  time      = 483638.75;

  interval = 300; % Linear interpolation (half) interval for sats.

  % select five satellites
  % Files that contain ephemeris constants.  These happened to be
  % visible at the data collection time.
  sv1 = sv7_eph ;
  sv2 = sv2_eph ;
  sv3 = sv9_eph ;
  sv4 = sv4_eph ;
  sv5 = sv12_eph ;

  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  %
  % Synthesize measurement vector
  %
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

35

```
% For now, ignore that satellites move during signal propagation
% time.  Thus, set ephemeris time to measurement time, time_true.
sv1(1) = time_true;
sv2(1) = time_true;
sv3(1) = time_true;
sv4(1) = time_true;
sv5(1) = time_true;

% Get satellite positions
sat(:,1) = satellite_ephemeris(sv1);
sat(:,2) = satellite_ephemeris(sv2);
sat(:,3) = satellite_ephemeris(sv3);
sat(:,4) = satellite_ephemeris(sv4);
sat(:,5) = satellite_ephemeris(sv5);

% Calculate perfect measured range differences.
for i = 1:4
  r_diff_t(i) = norm(sat(:,1) - U_true) - norm(sat(:,i+1) - U_true) ;
end

% Add noise to the measurements
light_speed = 3e8 ;                       % Meters/sec
dt_sigma = 1e-7 ;           % One tenth of a chip.
%  r_diff_m = r_diff_t + (light_speed * dt_sigma * randn(1,4)) ;
r_diff_m = r_diff_t + [0 0 0 10] ;
r_diff_m = r_diff_m' ;

dr_var = dt_sigma * 3e8 * dt_sigma * 3e8 ;
r_diff_covar = [        dr_var      0           0           0 ; ...
           0            dr_var      0           0 ; ...
           0            0           dr_var      0 ; ...
           0            0           0           dr_var ]

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Find satellite average velocities
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Set time to beginning of interpolation interval
% First element of svi is always time.
sv1(1) = time - interval;
sv2(1) = time - interval;
```

```
sv3(1) = time - interval;
sv4(1) = time - interval;
sv5(1) = time - interval;

% Get satellite positions
satA(:,1) = satellite_ephemeris(sv1);
satA(:,2) = satellite_ephemeris(sv2);
satA(:,3) = satellite_ephemeris(sv3);
satA(:,4) = satellite_ephemeris(sv4);
satA(:,5) = satellite_ephemeris(sv5);

% Set time to end of interpolation interval.
sv1(1) = time + interval;
sv2(1) = time + interval;
sv3(1) = time + interval;
sv4(1) = time + interval;
sv5(1) = time + interval;

% Get satellite positions.
satB(:,1) = satellite_ephemeris(sv1);
satB(:,2) = satellite_ephemeris(sv2);
satB(:,3) = satellite_ephemeris(sv3);
satB(:,4) = satellite_ephemeris(sv4);
satB(:,5) = satellite_ephemeris(sv5);

% calculate x,y,z velocities (in m/s)
V = (satB-satA)/interval;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Form initial H matrix of partials
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Partials taken at guessed time and user position.
% Need partials of sat coordinates w.r.t. time. Over the "interval"
% specified above we assume sat velocity is constant. Set partials
% to sat velocities. Compute vector and range from object to each
% satellite.

% Get satellite positions at guessed time.
sv1(1) = time ;
sv2(1) = time ;
```

```
sv3(1) = time ;
sv4(1) = time ;
sv5(1) = time ;
sat(:,1) = satellite_ephemeris(sv1);
sat(:,2) = satellite_ephemeris(sv2);
sat(:,3) = satellite_ephemeris(sv3);
sat(:,4) = satellite_ephemeris(sv4);
sat(:,5) = satellite_ephemeris(sv5);

% Compute estimated sat user vectors and ranges.
for i = 1:5,
  sat_U(:,i) = sat(:,i) - Ub ;
  range(i) = norm(sat_U(:,i)) ;
  DV(:,i) = sat_U(:,i) / range(i) ;
end

% Form H matrix of partials at estimated user location and time:
% i'th row of H is H = (pD1i/px, pD1i/py, pD1i/pz, pD1i/pt)
% First three elements is just the difference between the normalized
% direction vectors to the first and to the i'th satellites.
% Last element is difference between 1st and i'th inner products
% of direction vectors with satellite velocity vectors.
for i = 1:4,
  H(i,1:3) = DV(:,i+1)' - DV(:,1)' ;
  H(i,4) = DV(:,1)' * V(:,1) - DV(:,i+1)' * V(:,i+1) ;
end

% now compute the first estimate of G
%  G = inv(H);

loop = 0; epsilon = 1; %meter
while ( loop < 20)
  loop = loop + 1;
  % step 3

  % Substitute t into ephemeris data and get positions at guessed time.
  % Here we use actual ephemeris equations to find sat positions,
  % but note that the time derivatives of sat position are taken from
  % the linear interpolation.
  sv1(1) = time ;
  sv2(1) = time ;
  sv3(1) = time ;
  sv4(1) = time ;
```

```
sv5(1) = time ;
sat(:,1) = satellite_ephemeris(sv1);
sat(:,2) = satellite_ephemeris(sv2);
sat(:,3) = satellite_ephemeris(sv3);
sat(:,4) = satellite_ephemeris(sv4);
sat(:,5) = satellite_ephemeris(sv5);

sat1(:,loop) = sat(:,1) ;
sat2(:,loop) = sat(:,2) ;
sat3(:,loop) = sat(:,3) ;
sat4(:,loop) = sat(:,4) ;
sat5(:,loop) = sat(:,5) ;

% Compute vector and range from object to each satellite.
% Needed for new H matrix.
for i = 1:5,
  sat_U(:,i) = sat(:,i) - Ub ;
  range(i) = norm(sat_U(:,i)) ;
  DV(:,i) = sat_U(:,i) / range(i) ;
end

% compute r_diff using Ubar and t
r_diff(1) = range(1) - range(2) ;
r_diff(2) = range(1) - range(3) ;
r_diff(3) = range(1) - range(4) ;
r_diff(4) = range(1) - range(5) ;

% step 4
% compute range-difference error.
r_diff_error(:,loop) = r_diff_m - r_diff' ;

% step 5
% Form H matrix of partials:
% i'th row of H is H = (pD1i/px, pD1i/py, pD1i/pz, pD1i/pt)
% First three elements is just the difference between the normalized
% direction vectors to the first and to the i'th satellites.
% Last element is difference between 1st and i'th inner products
% of direction vectors with satellite velocity vectors.
for i = 1:4,
  H(i,1:3) = DV(:,i+1)' - DV(:,1)' ;
  H(i,4) = DV(:,1)' * V(:,1) - DV(:,i+1)' * V(:,i+1) ;
```

```
    end
    H4(:,loop) = H(:,4) ;

    % step 6
    % compute G via 1 iteration
%     G = G*(2*eye(4)-H*G);
    G = inv(H) ;
    G4(loop,:) = G(4,:) ;

    % step 7
    % compute dU as G * r_diff_error(:,loop)
    dU4 = G * r_diff_error(:,loop) ;
    dU = dU4(1:3);
    dt = dU4(4);

    % step 8
    % update position and time estimate Ub = Ub + dU, time = time + dt
    ans_vec(:,loop) = [Ub',time]' ;
    Ub = Ub + dU;
    time = time + dt;

    loop;
  end % while

  U_covar = G * r_diff_covar * G' ;
  U_sigma(1) = sqrt(U_covar(1,1)) ;
  U_sigma(2) = sqrt(U_covar(2,2)) ;
  U_sigma(3) = sqrt(U_covar(3,3)) ;
  U_sigma(4) = sqrt(U_covar(4,4)) ;

  % Plot x,y,z, or t convergence vs loop iteration number.
  % 1 -> x, 2 -> y, 3 -> z, 4 -> time.

  %         |
  %         |
  %         v

  plot(ans_vec(1,:)) ;
end


function [vec_satellite_ecf] = satellite_ephemeris(input_vector)

% The following code for calculating satellite positions from ephemeris
% data is taken from ICD-GPS-2000, table 20-IV.
```

```
t            = input_vector(1,1) ;
m_0          = input_vector(2,1) ;
delta_n      = input_vector(3,1) ;
e            = input_vector(4,1) ;
sqrt_a       = input_vector(5,1) ;
OMEGA_0      = input_vector(6,1) ;
i_0          = input_vector(7,1) ;
omega        = input_vector(8,1) ;
OMEGA_dot    = input_vector(9,1) ;
i_dot        = input_vector(10,1) ;
c_uc         = input_vector(11,1) ;
c_us         = input_vector(12,1) ;
c_rc         = input_vector(13,1) ;
c_rs         = input_vector(14,1) ;
c_ic         = input_vector(15,1) ;
c_is         = input_vector(16,1) ;
t_oe         = input_vector(17,1) ;
iode         = input_vector(18,1) ;

% Earth mass times universal gravity constant (see Ref 1, pp 34).
mu           = 3.986005e14 ;

% Earth rotation rate.
OMEGA_e_dot  = 7.2921151467e-5 ;

% Orbit ellipse semimajor axis length
a            = (sqrt_a)^2 ;

% Mean angular motion (Average angular rate of orbit).
n_0          = (mu/(a^3))^0.5 ;

% Time elapsed since ephemeris reference epoch.
t_k          = t - t_oe ;

if t_k > 302400
        t_k = t_k-604800 ; %correct time for EOW crossovers
elseif t_k < -302400
        t_k = t_k+604800 ; %correct time for EOW crossovers
end

% Corrected mean angular motion.
n            = n_0 + delta_n ;

% Mean anomaly (false sat orbit angle in orbit plane, using mean motion).
m_k          = m_0 + n*t_k ;
```

```
% Eccentric anomaly (true sat orbit angle in orbit plane, from ellipse center).
e_k                    = kepler(e,m_k,0.01,10000) ;

% Terms for finding true anomaly.
cosf_k                 = (cos(e_k) – e)/(1 – e*cos(e_k)) ;
sinf_k                 = ((1 – e^2)^0.5)*sin(e_k)/(1 – e*cos(e_k)) ;

% True anomaly (true sat orbit angle in orbit plane, from earth center).
f_k                    = atan2(sinf_k,cosf_k) ;

% Argument of lattitude (sat orbit angle from ascending node point on equator)
phi_k                  = f_k + omega ;

sin2phik       = sin(2*phi_k) ;
cos2phik       = cos(2*phi_k) ;

% Argument of lattitude correction.
delta_u_k      = c_us*sin2phik + c_uc*cos2phik ;

% Radius correction.
delta_r_k      = c_rc*cos2phik + c_rs*sin2phik ;

% Correction to inclination.
delta_i_k      = c_ic*cos2phik + c_is*sin2phik ;

% Corrected argument of lattitude.
u_k                    = phi_k + delta_u_k ;

% Corrected radius.
r_k                    = a*(1 – e*cos(e_k)) + delta_r_k ;

% Corrected inclination.
i_k                    = i_0 + delta_i_k + i_dot*t_k ;

% Final x–y position in orbital plane.
x_k_prime      = r_k*cos(u_k) ;
y_k_prime      = r_k*sin(u_k) ;

% Longitude of ascending node.
omega_k = OMEGA_0 + (OMEGA_dot–OMEGA_e_dot)*t_k – OMEGA_e_dot*t_oe ;

% Final sat position in earth–fixed coordinates.
x_k                    = x_k_prime*cos(omega_k) – y_k_prime*cos(i_k)*sin(omega_k) ;
y_k                    = x_k_prime*sin(omega_k) + y_k_prime*cos(i_k)*cos(omega_k) ;
z_k                    = y_k_prime*sin(i_k) ;
```

```
vec_satellite_ecf = [x_k;y_k;z_k] ;

end

function x = kepler(a,b,epsilon,N)

%KEPLER KEPLER(a,b,epsilon,N) returns the iterated solution to
%       x=a*sin(x)+b. Iterates until the new value is within epsilon
%       of the old value or until the number of iterations
%               reaches N.
old_x = 0;
new_x = b;
iterations=0;
while ((abs(new_x-old_x)>epsilon) & (iterations<N))
                old_x = new_x;
                %disp(new_x);
                new_x = a*sin(old_x)+b;
                iterations = iterations+1;
end
if iterations<N
                x=new_x;
                %disp('Iterations:');
                %disp(iterations);
                %disp(a);
                %disp(b);
                %disp(x);
elseif iterations==N
                disp('Fixed Point Iteration Failed');

end

function [param_vec] = sv7_eph()

EphemerisSV07length     = 167;%bytes
SV_PRN                          = 7;

t                               = 483634.754;

t_ephem                         = 483042;

weeknum                         = 763;
codeL2                          = 1;
L2Pdata                         = 0;
SVacc_raw               = 7;
```

```
SV_health   = 0;
IODC        = 120;
T_GD        = 1.39698e-09;
t_oc        = 485504;
a_f2        = 0;
a_f1        = 1.13687e-13;
a_f0        = 0.000697501;
SVacc       = 32;
IODE        = 120;
fit_intvl   = 0;
C_rs        = -17.2812;
delta_n     = 4.39733e-09;
M_0         = -1.6207;
C_uc        = -9.27597e-07;
ecc         = 0.00659284;
C_us        = 9.46783e-06;
sqrt_A      = 5153.78;
t_oe        = 485504;
C_ic        = 1.47149e-07;
OMEGA_0     = 0.793991;
C_is        = -9.31323e-09;
i_0         = 0.962958;
C_rc        = 197.938;
omega       = -2.69189;
OMEGADOT    = -7.93212e-09;
IDOT        = 7.82175e-11;
Axis        = 2.65614e+07;
n           = 0.00014585;
r1me2       = 0.999978;
OMEGA_n     = -34.6095;
ODOT_n      = -7.29291e-05;

param_vec = [t;
             M_0;
             delta_n;
             ecc;
             sqrt_A;
             OMEGA_0;
             i_0;
             omega;
             OMEGADOT;
```

```
                    IDOT;
                    C_uc;
                    C_us;
                    C_rc;
                    C_rs;
                    C_ic;
                    C_is;
                    t_oe;
                    IODE];

function [param_vec] = sv2_eph()

EphemerisSV02length   = 167;%bytes
SV_PRN                = 2;

t                     = 483634.754;

t_ephem               = 483042;

weeknum               = 763;
codeL2                = 1;
L2Pdata               = 0;
SVacc_raw             = 7;
SV_health             = 0;
IODC                  = 165;
T_GD                  = 9.31323e-10;
t_oc                  = 489600;
a_f2                  = 0;
a_f1                  = -2.16005e-12;
a_f0                  = -0.000109646;
SVacc                 = 32;
IODE                  = 165;
fit_intvl             = 0;
C_rs                  = -30.6562;
delta_n               = 4.92235e-09;
M_0                   = 0.0785892;
C_uc                  = -1.52551e-06;
ecc                   = 0.0134761;
C_us                  = 4.63426e-06;
sqrt_A                = 5153.66;
t_oe                  = 489600;
C_ic                  = 1.09896e-07;
OMEGA_0               = -0.26998;
```

```
C_is                      = 1.63913e-07;
i_0                       = 0.953377;
C_rc                      = 285.688;
omega                     = -2.63668;
OMEGADOT                  = -8.54964e-09;
IDOT                      = -2.16795e-10;
Axis                      = 2.65602e+07;
n                         = 0.00014586;
r1me2                     = 0.999909;
OMEGA_n                   = -35.9722;
ODOT_n                    = -7.29297e-05;

param_vec =     [t;
                 M_0;
                 delta_n;
                 ecc;
                 sqrt_A;
                 OMEGA_0;
                 i_0;
                 omega;
                 OMEGADOT;
                 IDOT;
                 C_uc;
                 C_us;
                 C_rc;
                 C_rs;
                 C_ic;
                 C_is;
                 t_oe;
                 IODE];

function [param_vec] = sv9_eph()

EphemerisSV09length    = 167;%bytes
SV_PRN                    = 9;

t                         = 483634.754;

t_ephem                   = 483042;

weeknum                   = 763;
codeL2                    = 1;
L2Pdata                   = 0;
```

```
SVacc_raw       = 7;
SV_health       = 0;
IODC                    = 592;
T_GD                    = 1.39698e-09;
t_oc                    = 489600;
a_f2                    = 0;
a_f1                    = -1.13687e-12;
a_f0                    = -5.16325e-06;
SVacc                   = 32;
IODE                    = 80;
fit_intvl       = 0;
C_rs                    = 80.1875;
delta_n                 = 4.85127e-09;
M_0                     = -2.65977;
C_uc                    = 4.25801e-06;
ecc                     = 0.00292443;
C_us                    = 5.34952e-06;
sqrt_A                  = 5153.73;
t_oe                    = 489600;
C_ic                    = 5.58794e-09;
OMEGA_0                 = -1.28659;
C_is                    = 2.23517e-08;
i_0                     = 0.950485;
C_rc                    = 270.656;
omega                   = -0.581175;
OMEGADOT                = -8.33535e-09;
IDOT                    = 2.4501e-10;
Axis                    = 2.6561e+07;
n                       = 0.000145854;
r1me2                   = 0.999996;
OMEGA_n                 = -36.9888;
ODOT_n                  = -7.29295e-05;

param_vec =     [t;
                        M_0;
                        delta_n;
                        ecc;
                        sqrt_A;
                        OMEGA_0;
                        i_0;
                        omega;
```

```
                    OMEGADOT;
                    IDOT;
                    C_uc;
                    C_us;
                    C_rc;
                    C_rs;
                    C_ic;
                    C_is;
                    t_oe;
                    IODE];

function [param_vec] = sv4_eph()

EphemerisSV04length     = 167;%bytes
SV_PRN                  = 4;

t                       = 483634.754;

t_ephem                 = 483048;

weeknum                 = 763;
codeL2                  = 1;
L2Pdata                 = 0;
SVacc_raw               = 7;
SV_health               = 0;
IODC                    = 711;
T_GD                    = 1.39698e-09;
t_oc                    = 489600;
a_f2                    = 0;
a_f1                    = 1.59162e-12;
a_f0                    = 4.03658e-05;
SVacc                   = 32;
IODE                    = 199;
fit_intvl               = 0;
C_rs                    = 15.4375;
delta_n                 = 4.67448e-09;
M_0                     = 2.57307;
C_uc                    = 9.76026e-07;
ecc                     = 0.00303051;
C_us                    = 6.10203e-06;
sqrt_A                  = 5153.62;
t_oe                    = 489600;
C_ic                    = 4.09782e-08;
```

```
OMEGA_0          = 1.86007;
C_is             = 1.11759e-08;
i_0              = 0.963886;
C_rc             = 261.562;
omega            = -1.24408;
OMEGADOT         = -8.10534e-09;
IDOT             = 2.62511e-10;
Axis             = 2.65598e+07;
n                = 0.000145863;
r1me2            = 0.999995;
OMEGA_n          = -33.8421;
ODOT_n           = -7.29293e-05;

param_vec =   [t;
               M_0;
               delta_n;
               ecc;
               sqrt_A;
               OMEGA_0;
               i_0;
               omega;
               OMEGADOT;
               IDOT;
               C_uc;
               C_us;
               C_rc;
               C_rs;
               C_ic;
               C_is;
               t_oe;
               IODE];

function [param_vec] = sv12_eph()

% This ephemeris data was extracted from test0826.asc
% by Glen Brooksby.

EphemerisSV12length   = 167;%bytes
SV_PRN           = 12;

t                = 483634.754;

t_ephem          = 482748;
```

```
weeknum          = 763;
codeL2           = 1;
L2Pdata          = 0;
SVacc_raw        = 2;
SV_health        = 0;
IODC             = 323;
T_GD             = 2.79397e-09;
t_oc             = 489600;
a_f2             = 0;
a_f1             = -1.21645e-11;
a_f0             = -2.44565e-05;
SVacc            = 4;
IODE             = 67;
fit_intvl        = 0;
C_rs             = 42.75;
delta_n          = 1.96937e-09;
m_0              = 3.01393;
C_uc             = 2.28174e-06;
ecc              = 0.0147704;
C_us             = 5.3402e-06;
sqrt_a           = 5153.5;
t_oe             = 489600;
C_ic             = -1.65775e-07;
omega_0          = -0.925903;
C_is             = 3.91155e-08;
i_0              = 1.08737;
C_rc             = 338.719;
omega            = -0.167314;
OMEGADOT         = -6.63885e-09;
IDOT             = 1.2572e-10;
Axis             = 2.65585e+07;
n                = 0.000145871;
r1me2            = 0.999891;
OMEGA_n          = -36.6281;
ODOT_n           = -7.29278e-05;

param_vec =   [t;
                 m_0;
                 delta_n;
                 ecc;
                 sqrt_a;
```

```
omega_0;
i_0;
omega;
OMEGADOT;
IDOT;
C_uc;
C_us;
C_rc;
C_rs;
C_ic;
C_is;
t_oe;
IODE];
```